CLAIMS

What is claimed is:

1. A method for detecting an error in an interaction between a plurality of software systems, comprising:

providing information about at least one of at least first and second software systems, and a mapping between at least a portion of said at least first and second software systems; and

examining said at least one of said first and second software systems and said mapping to determine an error in an interaction between said at least first and second software systems.

2. The method of claim 1, wherein said information provided about at least one software system of said at least first and second software systems includes a schema of said at least one software system.

3. The method of claim 1, wherein said information provided about at least one software system of said at least first and second software systems includes integrity constraints of the at least one software system.

4. The method of claim 1, wherein said information provided about at least one software system of said at least first and second software systems includes a unified modeling language (UML) model of said at least one software system.

5. The method of claim 1, wherein said information provided about at least one software system of said at least first and second software systems includes code of the at least one software system.

6. The method of claim 1, wherein said information provided about at least one software system of said at least first and second software systems includes at least one of:

a schema of said at least one software system;

integrity constraints of said at least one software system;

a specification of said at least one software system;

a unified modeling language (UML) model of said at least one software system; and

code of said at least one software system.

7. The method of claim 1, wherein said information provided about at least one software system of said at least first and second software systems includes information about a sub-component of said at least one software system.

8. The method of claim 1, wherein said information provided about at least

one software system of said at least first and second software systems includes information about less than an entirety of said at least one software system.

9. The method of claim 1, wherein one of said first and second software systems comprises a database.

10. The method of claim 9, wherein said information provided about said at least one software system of said at least first and second software systems includes schema information of said database.

11. The method of claim 9, wherein said information provided about said at least one software system of said at least first and second software systems comprises information about values in said database.

12. The method of claim 1, wherein one of said first and second software systems comprises an application.

13. The method of claim 12, wherein said information provided about said one of said first and second software systems includes programming language types.

14. The method of claim 1, wherein one of said first and second software systems comprises one of an extensible markup language (XML) repository and an XML database.

15. The method of claim 14, wherein said information provided about said one of said first and second software systems includes one of XML schema information and XML document type definition (DTD) information.

16. The method of claim 2, wherein said schema includes XML schema information.

17. The method of claim 1, wherein said mapping is provided explicitly.

18. The method of claim 1, wherein said mapping is inferred from said information about said at least first and second software systems.

19. The method of claim 1, wherein said error comprises an integrity constraint violation.

20. The method of claim 1, wherein said error comprises a potential error representing a warning that an error may occur.

21. The method of claim 1, wherein said error comprises a definite error representing one of that an error will occur and that an error has occurred.

22. The method of claim 1, wherein said error is found prior to said interaction between the at least first and second software systems.

23. The method of claim 1, wherein said error is found during said run time of the at least first and second software systems.

24. The method of claim 1, wherein said error is found prior to any of said interaction between the at least first and second software systems, and during said runtime of the at least first and second software systems.

25. The method of claim 1, further comprising:

inserting a check into at least one of said at least first and second software systems.

26. The method of claim 25, wherein said check is inserted at a location directed by a programmer.

27. The method of claim 25, wherein said at least first and second software systems are checked after an interaction therebetween.

28. The method of claim 25, wherein said at least first and second software systems are checked before an interaction therebetween.

29. The method of claim 25, wherein said at least first and second software systems are checked prior to an end of a transaction.

30. The method of claim 25, further comprising:

performing static analysis of said at least first and second software systems to at least one of simplify, eliminate, and approximate said check.

31. The method of claim 1, further comprising:

performing static analysis of said at least one of at least first and second software systems.

32. The method of claim 3, further comprising:

representing said integrity constraints of said at least first and second software systems in a common constraint model

33. The method of claim 32, further comprising:

analyzing said integrity constraints in said common constraint model.

34. The method of claim 33, further comprising:

based on said analyzing, if an inconsistency is detected, then outputting an error.

35. The method of claim 31, further comprising modifying said integrity constraints in said common constraint model.

36. The method of claim 31, further comprising:

generating a check from said integrity constraints.

37. The method of claim 31, further comprising:

providing a shadow database in one of said at least first and second software systems,

said shadow database containing partial knowledge of the other of said at least first and second software systems and being used to perform a check.

38. The method of claim 37, wherein said partial knowledge includes partial knowledge of data values in said other of said at least first and second software systems.

39. The method of claim 37, wherein said partial knowledge includes partial knowledge of non-existence of data values in said other of said at least first and second software systems.

40. The method of claim 1, further comprising:

reporting said error.

41. The method of claim 40, wherein said reporting comprises notifying before running at least one of said at least first and second software systems.

42. The method of claim 40, wherein said reporting comprises notifying while running at least one of said at least first and second software systems.

43. The method of claim 40, wherein said reporting allows said one of said at least first and second software systems to address said error.

44. The method of claim 40, wherein said reporting suggests how said error may be addressed.

45. A method of detecting an error in a database interaction, comprising:

examining database code for database constraints;

examining application code for application-level constraints; and

analyzing a mapping between said database code and said application code, to determine an error in a database interaction.

46. The method of claim 45, further comprising:

generating a check in said application code for enforcing said database and application-level constraints.

47. The method of claim 45, further comprising:

forming a shadow database in said application code representing a portion of said database.

48. The method of claim 45, wherein said error comprises an integrity constraint violation.

49. The method of claim 45, further comprising:

inputting said database constraints and said application-level constraints, and said mapping into a common constraint model.

50. The method of claim 49, further comprising:

before a program including said application and interacting with said database, is run and after said inputting, performing a static analysis to identify locations of where an error may arise.

51. The method of claim 45, further comprising:

after identifying said error and prior to running a program including said application, raising a notification.

52. The method of claim 45, wherein, at runtime of said program, when an error is detected as occurring, raising a notification.

53. A method of detecting an integrity constraint violation in a database interaction, comprising:

examining a database schema;

examining an application type; and

analyzing a mapping between said database schema and said application type, to determine whether an integrity constraint violation will occur in said database interaction with said application.

54. The method of claim 53, wherein said database schema provides each of the integrated constraints defined in the database, and

wherein said application type includes application code including integrity constraints defined therein.

55. The method of claim 54, further comprising:

generating a check in said application code for enforcing said database and application-level constraints.

56. The method of claim 53, further comprising:

forming a shadow database in application code representing a portion of said database.

57. The method of claim 53, further comprising:

inputting said database schema, said application type, and said mapping into a common constraint model.

58. The method of claim 57, further comprising:

before a program including said application and interacting with said database, is run and after said inputting, performing a static analysis to identify locations of where said integrity constraint violation may arise.

59. The method of claim 53, further comprising:

after determining said integrity constraint violation and prior to running a program including said application, raising a notification.

60. The method of claim 53, wherein, at runtime of said program, when an integrity constraint violation is determined to occur, raising a notification.

61. The method of claim 53, further comprising:

analyzing a common constraint model receiving said database schema, application type, and mapping, to determine an inconsistency between said database schema and said application type.

62. The method of claim 61, wherein if no said inconsistency is determined, then taking all of the common constraints and analyzing the application code with respect to the common constraints for an error in the application code.

63. The method of claim 62, wherein if no said error is determined in the application code, then inserting a check into said application code to enforce the constraints at runtime.

64. A system of detecting an error in a database interaction, comprising:

a module for examining a database code for database constraints;

a module for examining an application code for application-level constraints; and

an analyzing unit for analyzing a mapping between said database code and said application code, to determine an error in a database interaction.

65. A system for detecting an integrity constraint violation in a database interaction, comprising:

a common constraint model for analyzing database schema, application type, and a mapping between said database schema and said application type; and

a determining unit for determining whether an integrity constraint violation will occur in said database interaction with said application.

66. A method of constructing a program, comprising:

detecting, in an application, portions of said application code that will or may raise a database integrity constraint violation during an application-database interaction during runtime, the detecting including examining database schema, examining application type, and a mapping between the database schema and the application type;

inserting an integrity check notifying a programmer of such a definite or potential violation; and

completing the program.

67. A signal-bearing medium tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus to perform a method of claim 45.

68. A signal-bearing medium tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus to perform a method of claim 53.

69. The method of claim 1, wherein said information provided about at least one software system of said at least first and second software systems includes a specification of said at least one software system.